Nim Standard Library 1.3.5

Andreas Rumpf

August 21, 2020

Contents

Nim's library is divided into *pure libraries*, *impure libraries* and *wrappers*.

Pure libraries do not depend on any external `*.dll` or `lib*.so` binary while impure libraries do. A wrapper is an impure library that is a very low-level interface to a C library.

Read this document for a quick overview of the API design.

1 Nimble

Nim's standard library only covers the basics, check out https://nimble.directory/ for a list of 3rd party packages.

2 Pure libraries

2.1 Automatic imports

- system Basic procs and operators that every program needs. It also provides IO facilities for reading and writing text and binary files. It is imported implicitly by the compiler. Do not import it directly. It relies on compiler magic to work.
- threads Basic Nim thread support. **Note**: This is part of the system module. Do not import it explicitly. Enabled with `-threads:on`.
- channels Nim message passing support for threads. **Note**: This is part of the system module. Do not import it explicitly. Enabled with `-threads:on`.

2.2 Core

- bitops Provides a series of low level methods for bit manipulation.
- cpuinfo This module implements procs to determine the number of CPUs / cores.
- endians This module contains helpers that deal with different byte orders.
- lenientops Provides binary operators for mixed integer/float expressions for convenience.
- locks Locks and condition variables for Nim.
- macros Contains the AST API and documentation of Nim for writing macros.
- rlocks Reentrant locks for Nim.
- typeinfo Provides (unsafe) access to Nim's run time type information.
- typetraits This module defines compile-time reflection procs for working with types.
- volatile This module contains code for generating volatile loads and stores, which are useful in embedded and systems programming.

2.3 Algorithms

- algorithm Implements some common generic algorithms like sort or binary search.
- sequtils This module implements operations for the built-in seq type which were inspired by functional programming languages.

2.4 Collections

- critbits This module implements a *crit bit tree* which is an efficient container for a sorted set of strings, or for a sorted mapping of strings.
- deques Implementation of a double-ended queue. The underlying implementation uses a `seq`.
- heapqueue Implementation of a heap data structure that can be used as a priority queue.
- intsets Efficient implementation of a set of ints as a sparse bit set.
- lists Nim linked list support. Contains singly and doubly linked lists and circular lists ("rings").
- options The option type encapsulates an optional value.
- sets Nim hash and bit set support.
- sharedlist Nim shared linked list support. Contains shared singly linked list.
- sharedtables Nim shared hash table support. Contains shared tables.
- tables Nim hash table support. Contains tables, ordered tables and count tables.

2.5 String handling

- cstrutils Utilities for `cstring` handling.
- std/editdistance This module contains an algorithm to compute the edit distance between two Unicode strings.
- encodings Converts between different character encodings. On UNIX, this uses the `iconv` library, on Windows the Windows API.
- parseutils This module contains helpers for parsing tokens, numbers, identifiers, etc.
- pegs This module contains procedures and operators for handling PEGs.
- punycode Implements a representation of Unicode with the limited ASCII character subset.
- ropes This module contains support for a *rope* data type. Ropes can represent very long strings efficiently; especially concatenation is done in O(1) instead of O(n).
- strformat Macro based standard string interpolation / formatting. Inspired by Python's `f`-strings.
- strmisc This module contains uncommon string handling operations that do not fit with the commonly used operations in strutils.
- strscans This module contains a `scanf` macro for convenient parsing of mini languages.
- strtabs The `strtabs` module implements an efficient hash table that is a mapping from strings to strings. Supports a case-sensitive, case-insensitive and style-insensitive modes.
- strutils This module contains common string handling operations like changing case of a string, splitting a string into substrings, searching for substrings, replacing substrings.
- unicode This module provides support to handle the Unicode UTF-8 encoding.
- unidecode It provides a single proc that does Unicode to ASCII transliterations. Based on Python's Unidecode module.
- std/wordwrap This module contains an algorithm to wordwrap a Unicode string.

2.6 Time handling

- std/monotimes The `monotimes` module implements monotonic timestamps.
- times The `times` module contains support for working with time.

2.7 Generic Operating System Services

- distros This module implements the basics for OS distribution ("distro") detection and the OS's native package manager. Its primary purpose is to produce output for Nimble packages, but it also contains the widely used **Distribution** enum that is useful for writing platform specific code. See packaging for hints on distributing Nim using OS packages.
- dynlib This module implements the ability to access symbols from shared libraries.
- marshal Contains procs for serialization and deserialization of arbitrary Nim data structures.
- memfiles This module provides support for memory mapped files (Posix's `mmap`) on the different operating systems.
- os Basic operating system facilities like retrieving environment variables, reading command line arguments, working with directories, running shell commands, etc.
- osproc Module for process communication beyond `os.execShellCmd`.
- streams This module provides a stream interface and two implementations thereof: the `FileStream` and the `StringStream` which implement the stream interface for Nim file objects (`File`) and strings. Other modules may provide other implementations for this standard stream interface.
- terminal This module contains a few procedures to control the *terminal* (also called *console*). The implementation simply uses ANSI escape sequences and does not depend on any other module.

2.8 Math libraries

- complex This module implements complex numbers and their mathematical operations.
- fenv Floating-point environment. Handling of floating-point rounding and exceptions (overflow, zero-divide, etc.).
- math Mathematical operations like cosine, square root.
- mersenne Mersenne twister random number generator.
- random Fast and tiny random number generator.
- rationals This module implements rational numbers and their mathematical operations.
- stats Statistical analysis
- std/sums Fast summation functions.

2.9 Internet Protocols and Support

- asyncdispatch This module implements an asynchronous dispatcher for IO operations.
- asyncfile This module implements asynchronous file reading and writing using `asyncdispatch`.
- asyncftpclient This module implements an asynchronous FTP client using the `asyncnet` module.
- asynchttpserver This module implements an asynchronous HTTP server using the `asyncnet` module.
- asyncnet This module implements asynchronous sockets based on the `asyncdispatch` module.

- asyncstreams This module provides `FutureStream` - a future that acts as a queue.
- cgi This module implements helpers for CGI applications.
- cookies This module contains helper procs for parsing and generating cookies.
- httpclient This module implements a simple HTTP client which supports both synchronous and asynchronous retrieval of web pages.
- mimetypes This module implements a mimetypes database.
- nativesockets This module implements a low-level sockets API.
- net This module implements a high-level sockets API. It replaces the `sockets` module.
- selectors This module implements a selector API with backends specific to each OS. Currently epoll on Linux and select on other operating systems.
- smtp This module implement a simple SMTP client.
- uri This module provides functions for working with URIs.

2.10 Threading

- threadpool Implements Nim's spawn.

2.11 Parsers

- htmlparser This module parses an HTML document and creates its XML tree representation.
- json High performance JSON parser.
- lexbase This is a low level module that implements an extremely efficient buffering scheme for lexers and parsers. This is used by the diverse parsing modules.
- parsecfg The `parsecfg` module implements a high performance configuration file parser. The configuration file's syntax is similar to the Windows `.ini` format, but much more powerful, as it is not a line based parser. String literals, raw string literals and triple quote string literals are supported as in the Nim programming language.
- parsecsv The `parsecsv` module implements a simple high performance CSV parser.
- parseopt The `parseopt` module implements a command line option parser.
- parsesql The `parsesql` module implements a simple high performance SQL parser.
- parsexml The `parsexml` module implements a simple high performance XML/HTML parser. The only encoding that is supported is UTF-8. The parser has been designed to be somewhat error correcting, so that even some "wild HTML" found on the Web can be parsed with it.

2.12 Docutils

- packages/docutils/highlite Source highlighter for programming or markup languages. Currently only few languages are supported, other languages may be added. The interface supports one language nested in another.
- packages/docutils/rst This module implements a reStructuredText parser. A large subset is implemented. Some features of the markdown wiki syntax are also supported.
- packages/docutils/rstast This module implements an AST for the reStructuredText parser.
- packages/docutils/rstgen This module implements a generator of HTML/Latex from reStructuredText.

2.13 XML Processing

- xmltree A simple XML tree. More efficient and simpler than the DOM. It also contains a macro for XML/HTML code generation.
- xmlparser This module parses an XML document and creates its XML tree representation.

2.14 Generators

- htmlgen This module implements a simple XML and HTML code generator. Each commonly used HTML tag has a corresponding macro that generates a string with its HTML representation.

2.15 Hashing

- base64 This module implements a base64 encoder and decoder.
- hashes This module implements efficient computations of hash values for diverse Nim types.
- md5 This module implements the MD5 checksum algorithm.
- oids An OID is a global ID that consists of a timestamp, a unique counter and a random value. This combination should suffice to produce a globally distributed unique ID. This implementation was extracted from the Mongodb interface and it thus binary compatible with a Mongo OID.
- std/sha1 This module implements a sha1 encoder and decoder.

2.16 Miscellaneous

- browsers This module implements procs for opening URLs with the user's default browser.
- colors This module implements color handling for Nim.
- coro This module implements experimental coroutines in Nim.
- logging This module implements a simple logger.
- segfaults Turns access violations or segfaults into a `NilAccessDefect` exception.
- sugar This module implements nice syntactic sugar based on Nim's macro system.
- unittest Implements a Unit testing DSL.
- std/varints Decode variable length integers that are compatible with SQLite.

2.17 Modules for JS backend

- asyncjs Types and macros for writing asynchronous procedures in JavaScript.
- dom Declaration of the Document Object Model for the JS backend.
- jsconsole Wrapper for the `console` object.
- jscore Wrapper of core JavaScript functions. For most purposes you should be using the `math`, `json`, and `times` stdlib modules instead of this module.
- jsffi Types and macros for easier interaction with JavaScript.

3 Impure libraries

3.1 Regular expressions

- re This module contains procedures and operators for handling regular expressions. The current implementation uses PCRE.

3.2 Database support

- db_postgres A higher level PostgreSQL database wrapper. The same interface is implemented for other databases too.
- db_mysql A higher level MySQL database wrapper. The same interface is implemented for other databases too.
- db_sqlite A higher level SQLite database wrapper. The same interface is implemented for other databases too.

4 Wrappers

The generated HTML for some of these wrappers is so huge that it is not contained in the distribution. You can then find them on the website.

4.1 Windows specific

- winlean Contains a wrapper for a small subset of the Win32 API.
- registry Windows registry support.

4.2 UNIX specific

- posix Contains a wrapper for the POSIX standard.
- posix_utils Contains helpers for the POSIX standard or specialized for Linux and BSDs.

4.3 Regular expressions

- pcre Wrapper for the PCRE library.

4.4 GUI libraries

- iup Wrapper of the IUP GUI library.

4.5 Database support

- postgres Contains a wrapper for the PostgreSQL API.
- mysql Contains a wrapper for the mySQL API.
- sqlite3 Contains a wrapper for SQLite 3 API.
- odbcsql interface to the ODBC driver.

4.6 Network Programming and Internet Protocols

- openssl Wrapper for OpenSSL.